CSE 559A: Computer Vision

[credit: danjodon.deviantart.com]

Fall 2017: T-R: 11:30-1pm @ Lopata 101

Instructor: Ayan Chakrabarti (ayan@wustl.edu).
Staff: Abby Stylianou (abby@wustl.edu), Jarett Gross (jarett@wustl.edu)

http://www.cse.wustl.edu/~ayan/courses/cse559a/

Oct 31, 2017

GENERAL

- Project proposals due today. Submit on Blackboard.

- PSET 4 will be posted today. Due two weeks from now.

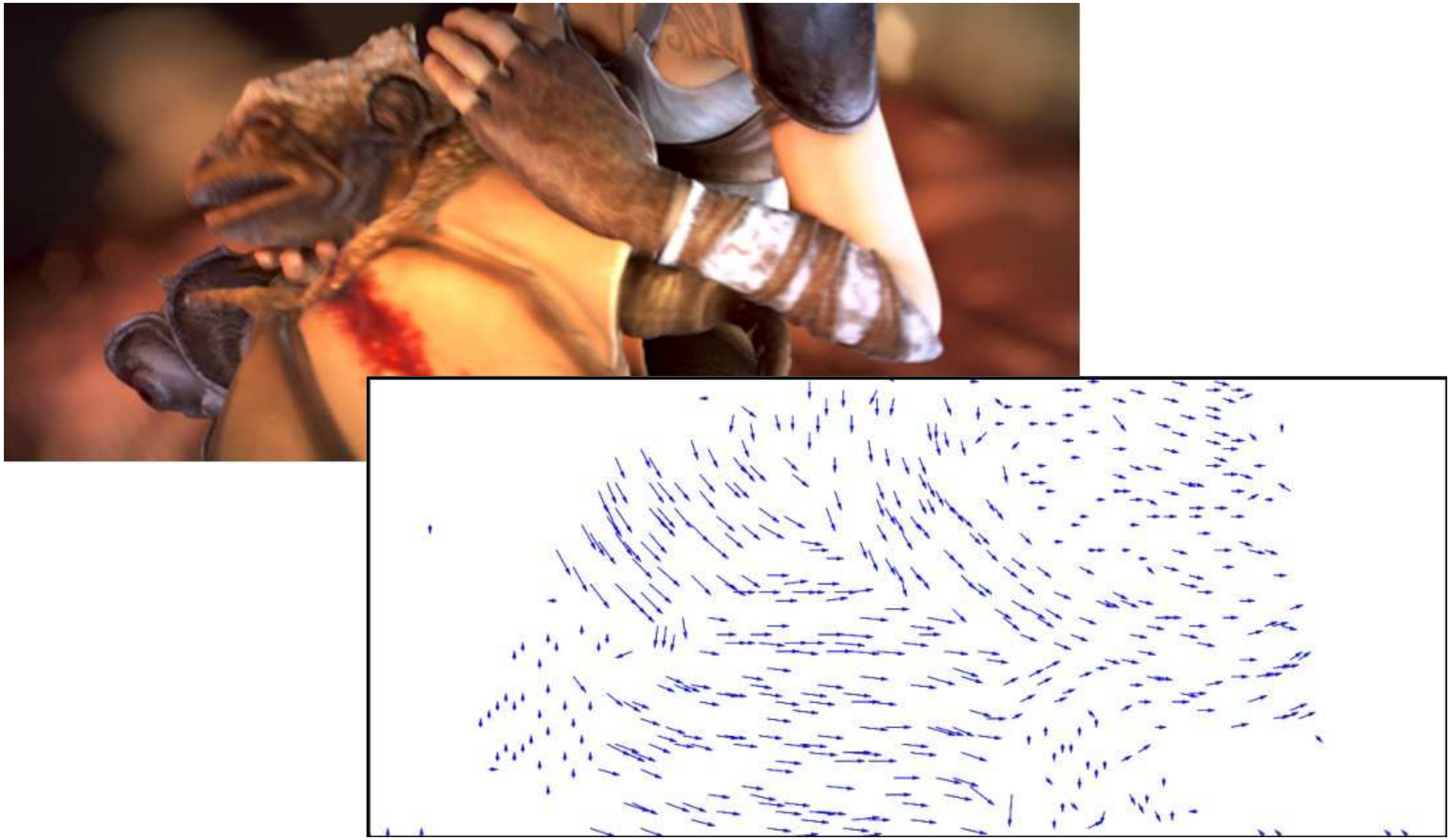
OPTICAL FLOW

Slides/ Examples from Steve Seitz, Lana Lazebnik, Subhransu Maji, Yasu Furukawa, The Sintel Dataset from MPI

OPTICAL FLOW

Optical Flow = How did the pixels move in the image plane between two frames ?

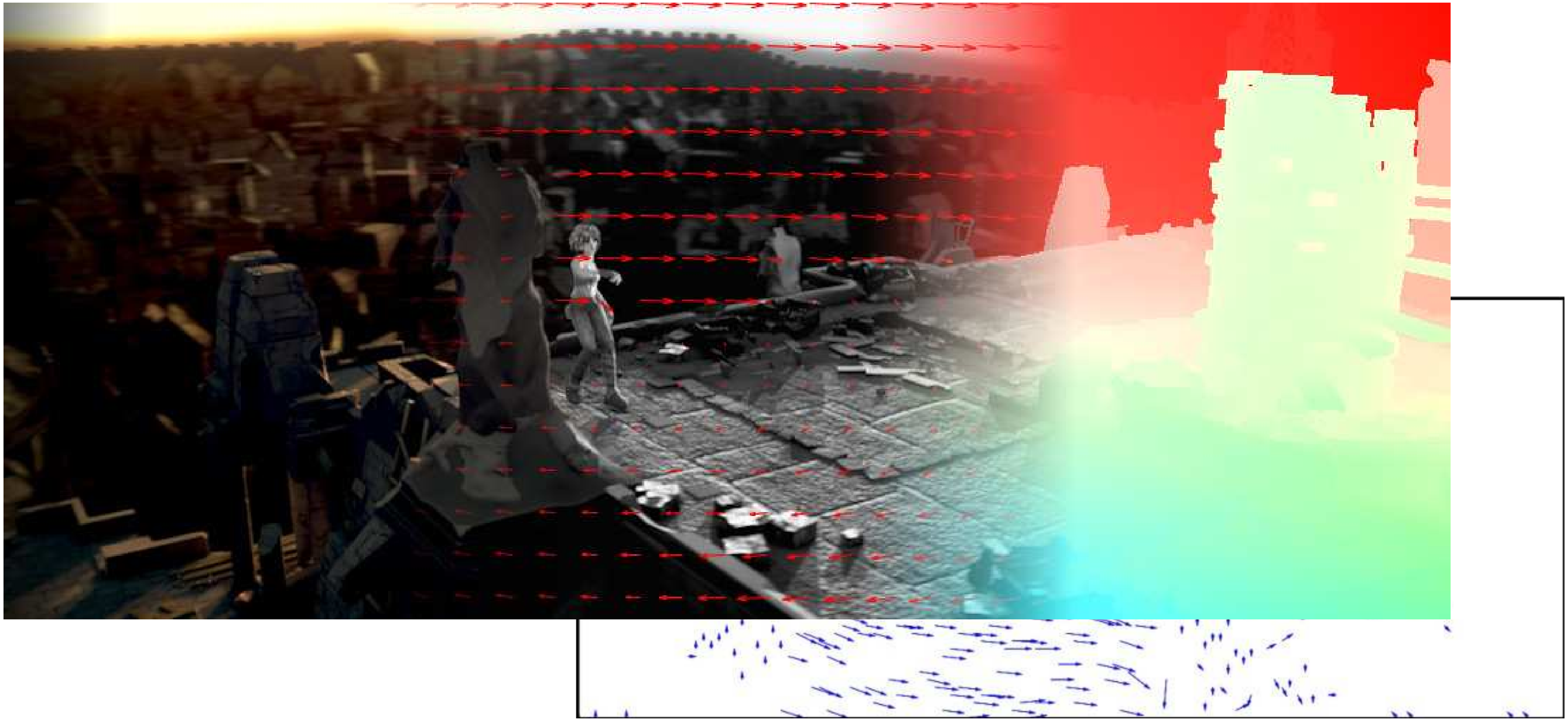
$$I_t[x, y] \rightarrow I_{t+1}\left[x + u[x, y], y + v[x, y]\right]$$



OPTICAL FLOW

Useful for:

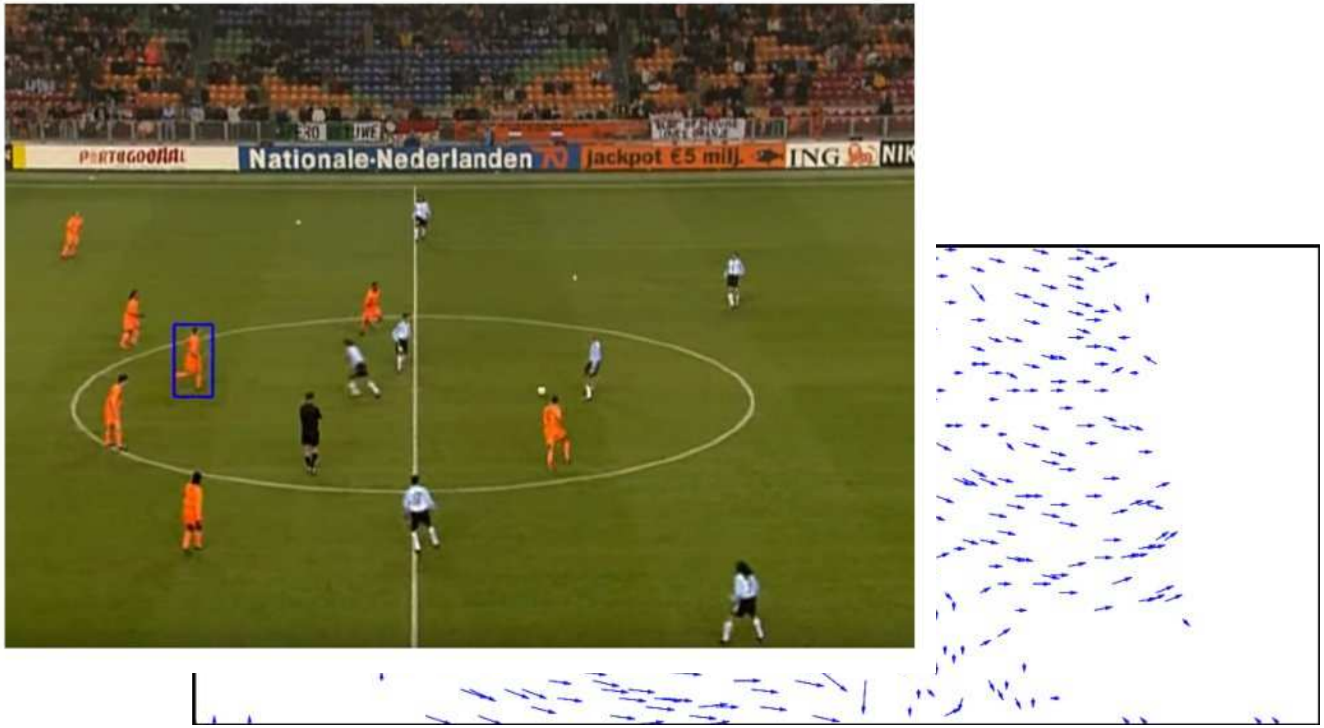
Analyzing motion / shape



OPTICAL FLOW

Useful for:

- Analyzing motion / shape
- Tracking Objects / People across time



OPTICAL FLOW

Useful for:

Analyzing motion / shape

Tracking Objects / People across time

Image Morphing

$$I_t[x, y] \rightarrow I_{t+1}\left[x + u[x, y], y + v[x, y]\right]$$

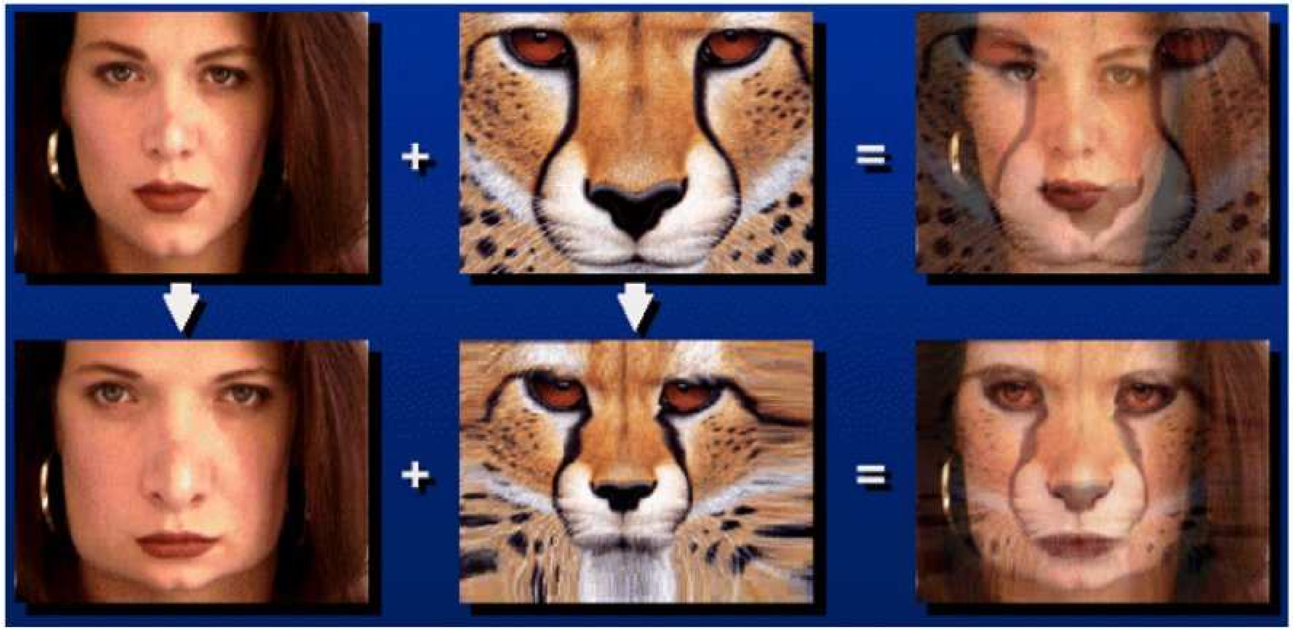
$$I_t[x + u[x, y]/2, y + v[x, y]/2]$$

Forms a mid-point of the deformation between two ~~frames~~ ...
images

OPTICAL FLOW

Useful for:

- Analyzing motion / shape
- Tracking Objects / People across time
- Image Morphing



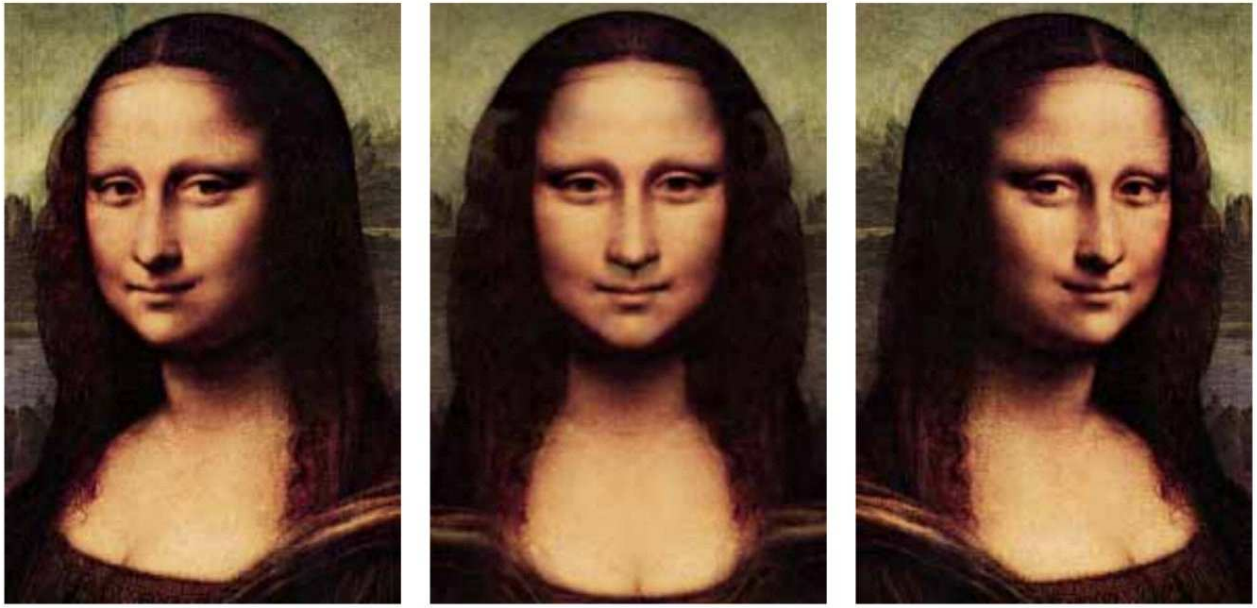
OPTICAL FLOW

Useful for:

Analyzing motion / shape

Tracking Objects / People across time

Image Morphing



OPTICAL FLOW

$$I_t[x, y] \rightarrow I_{t+1}\left[x + u[x, y], y + v[x, y]\right]$$

Correspondence search, without the benefit of epipolar geometry.

OPTICAL FLOW

Let's try to solve it assuming $u[x, y]$, $v[x, y]$ are very small. (Very little movement).

Also assume "brightness constancy":

$$I[x, y, t] = I[x + u[x, y], y + v[x, y], t + 1]$$

Do a Taylor approximation to "linearize" the RHS.

$$I[x, y, t] = I[x + u, y + v, t + 1] \approx I[x, y, t + 1] + \frac{\partial}{\partial x} I[x, y, t]u + \frac{\partial}{\partial y} I[x, y, t]v$$

$$\frac{\partial}{\partial_t} I[x, y, t] + \frac{\partial}{\partial x} I[x, y, t]u + \frac{\partial}{\partial y} I[x, y, t]v \approx 0$$

$$I_t + \langle [I_x, I_y], [u, v] \rangle = 0$$

Lucas-Kanade Method

OPTICAL FLOW

Lucas-Kanade Method

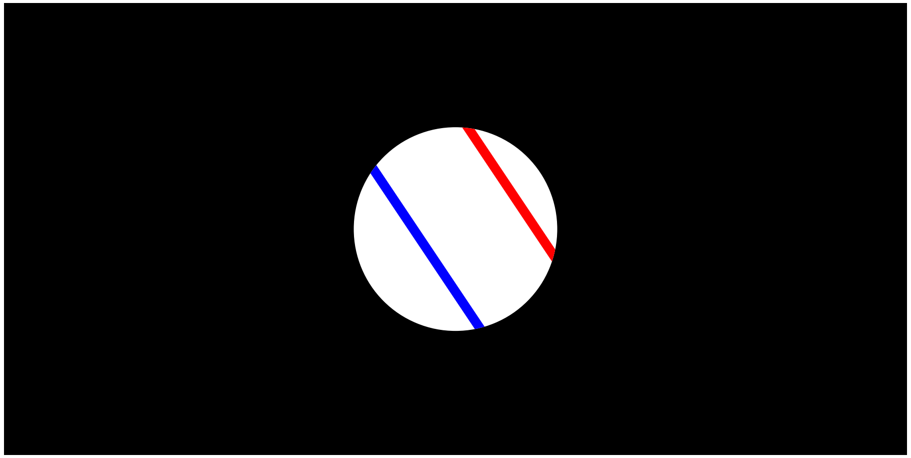
$$I_t + \langle [I_x, I_y], [u, v] \rangle = 0$$

- I_t is an image-shaped array of time gradients (subtracting $I[x, y, t+1] - I[x, y, t]$)
  - (Not to be confused with I_t notation from the last slide)
- I_x and I_y are x- and y- gradient images (e.g., use Sobel filters).
  - Often, you want to apply these on $\frac{I[x,y,t+1]+I[x,y,t]}{2}$

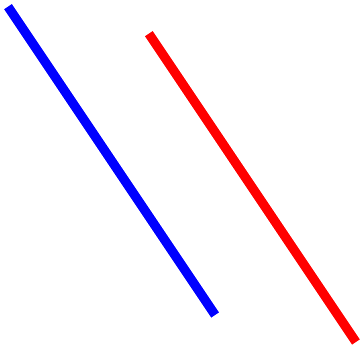
This is an equation on $u[x, y]$, $v[x, y]$ at each pixel location.

But one equation for two variables. Doesn't tell us about flow vector in direction "orthogonal" to image gradient.

Aperture Problem



Aperture Problem



OPTICAL FLOW

Aperture Problem

http://en.wikipedia.org/wiki/Barberpole_illusion

OPTICAL FLOW

Lucas-Kanade Method

Solution: Assume u, v is constant in a region, and get multiple equations.

So for $u[x, y] = u, v[x, y] = u$, consider a bunch of x', y' in a window around x, y.

$$I_x[x', y']\, u + I_y[x', y']\, v = -I_t[x', y']$$

Multiple equations, two variables: solve in the least squares sense.

$$\begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}$$

OPTICAL FLOW

Lucas-Kanade Method

$$\begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}$$

Summations are in a window around x, y.

How would you do this without looping over pixels ?

- Compute $I_x^2, I_x I_y, I_y^2, I_t I_x, \ldots$ point-wise.
- Use convolutions to do local summation.
- Form each element of left matrix and right vector as a separate image.
- Invert by pointwise operations on these images.

OPTICAL FLOW

Lucas-Kanade Method

$$\begin{bmatrix} \sum I_x^2 + \epsilon & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 + \epsilon \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}$$

Summations are in a window around x, y.

When will you get good answers and when will you get bad answers ?

In other words, when is the matrix invertible ?

- Matrix will be all zero in a smooth region.
- Matrix will be rank 1 if all gradients in one direction.
- Good when you have general texture.

For stability, add a small value to the diagonal elements of the matrix.

OPTICAL FLOW

Lucas-Kanade Method

Pyramid / Hierarchical Variant to handle large displacements

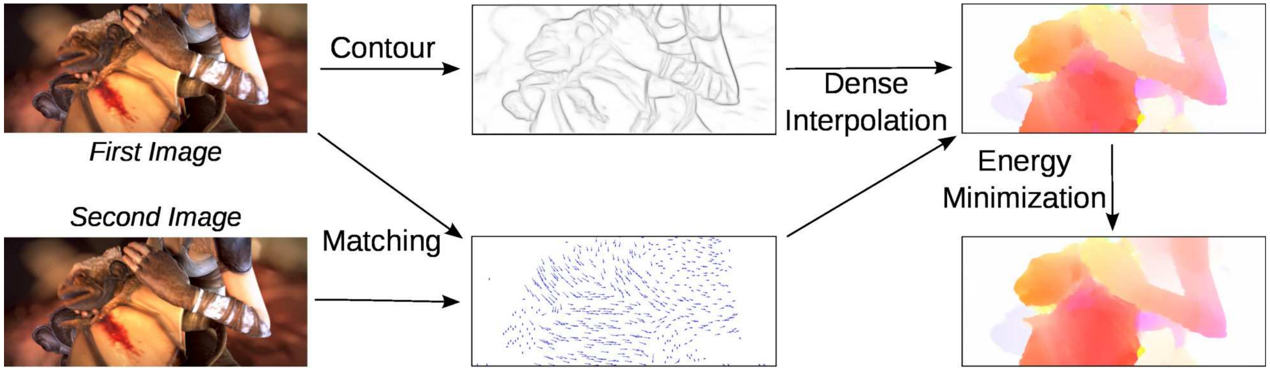
- First downsample images, solve it at a coarser scale.
- Then, upsample the flow-field, and compute displacement from that flow field.

So basically, warp image I_{t+1} based on flow-field from coarser level, and now find differential motion beyond that.

OPTICAL FLOW

State-of-the Art Methods

- Use energy minimization, complex features, contours, . . .

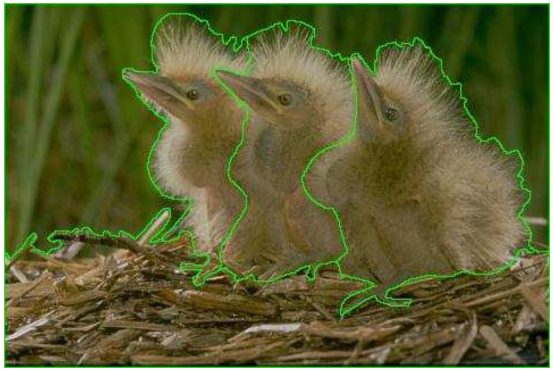


Jerome Revaud, Philippe Weinzaepfel, Zaid Harchaoui and Cordelia Schmid
EpicFlow: Edge-Preserving Interpolation of Correspondences for Optical Flow
CVPR 2015.

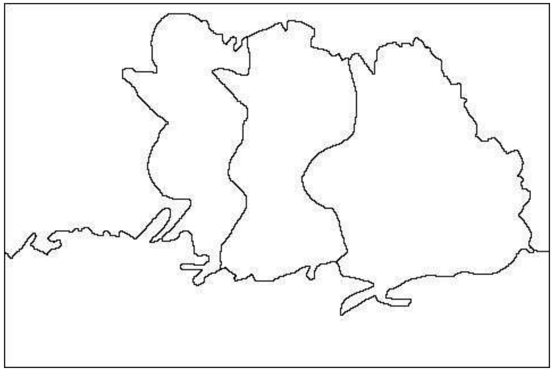
http://sintel.is.tue.mpg.de/results - For a leaderboard of best results.

GROUPING & SEGMENTATION

Partition the set of pixels into disjoint sets or groups



Dual of the edge detection problem !



GROUPING & SEGMENTATION

But what is the basis of this grouping ?

- Physical
  - Lie on the same surface / plane
  - Made of the same material
  - Moving together rigidly
- Semantic
  - Same object
  - Foreground / background
  - Interesting / non-interesting

Semantic segmentation: often humans will disagree on what goes where.

GROUPING & SEGMENTATION

Simplest Version: Superpixel Segementation

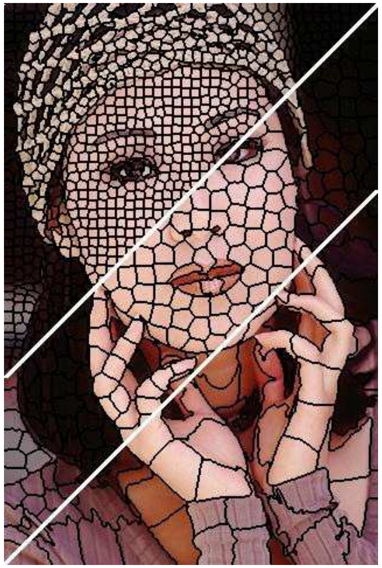
- Partition Image into a large number of segments called superpixels.
- Many segments, each segment relatively small.
- Oversegmentation of the image
  - Each object / plane / surface might be broken into multiple segments
  - But (hope) each segment does not cross a boundary.

- Can be based on appearance alone

- Simplifies further processing (dealing with K segments instead of N pixels)

GROUPING & SEGMENTATION

Image



Superpixels
(different levels)

GROUPING & SEGMENTATION

SLIC Superpixels

Achanta et al., 2010. Simple Linear Iterative Clustering.

Formally, given an image $I[n]$ with N pixels, you want group the pixels into $K << N$ super pixels.

You want to determine a label $L[n] \in \{1, 2, \dots K\}$ for every pixel n, based on some metric.

Note the value of L doesn't matter. What matters is similar pixels have the same label. This is clustering !

GROUPING & SEGMENTATION

SLIC Superpixels

Define an "augmented" image $I'[n]$ where each $I'[n] \in \mathbb{R}^5$

- First 3 dimensions are R,G,B
- Two dimensions are x and y co-ordinates.

For grayscale images, $I'[n] \in \mathbb{R}^3$.

GROUPING & SEGMENTATION

SLIC Superpixels

Determine labeling $L[n]$ to minimize the following cost:

$$L = \arg\min_{L} \min_{\{\mu_k\}} \sum_{k=1}^{K} \sum_{n:L[n]=k} \|I'[n] - \mu_k\|^2$$

Here, each $\mu_k \in \mathbb{R}^5$.

- This is K-means clustering.
- Easy to see that μ_k will be the mean of the I' vectors of pixels assigned to label k.
- We're saying that all pixels assigned the label k should be
  close to each other in the squared distance sense of their augmented vectors.
- This augmented vector encodes both appearance and location.
- So we want pixels that look the same and are close-by to have the same label.

GROUPING & SEGMENTATION

SLIC Superpixels

$$L = \arg\min_{L} \min_{\{\mu_k\}} \sum_{k=1}^{K} \sum_{n:L[n]=k} \|I'[n] - \mu_k\|^2$$

- Typically, use Lab color space instead of RGB.
- You can weight the contribution of location vs appearance by normalizing (x, y) in I' differently.
$$I'[n] = [I[n]_R, I[n]_G, I[n]_B, \alpha n_x, \alpha n_y]^T$$

GROUPING & SEGMENTATION

$$L = \arg\min_{L} \min_{\{\mu_k\}} \sum_{k=1}^{K} \sum_{n:L[n]=k} \|I'[n] - \mu_k\|^2$$

K-Means: Lloyd's algorithm

- Begin with some initial assignment $L[n]$ (more later).
- At each iteration ...

Step 1: For each k, assign

$$\mu_k = \text{Mean}\{I'[n]\}_{L[n]=k}$$

Step 2: For each n, assign

$$L[n] = \arg\min_{k} \|I'[n] - \mu_k\|^2$$

- Does this converge ?

But kind of expensive ? In our case, K and N will both be large.

GROUPING & SEGMENTATION

Next time:

- How to initialize $L[n]$, restrict search space of minimization.
- Other kinds of segmentation.